ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 2(31)

APPLICATION OF MULTISTART TABU SEARCH TO THE MAX-CUT PROBLEM

Gintaras Palubeckis

Department of Practical Informatics, Kaunas University of Technology
Studentu 50, LT-3031, Kaunas, Lithuania

Vita Krivickiene

Technical College of Kaunas
Tvirtoves al. 35, LT-3040, Kaunas, Lithuania

Abstract. In this paper, we investigate two multistart tabu search implementations for the MAX-CUT problem: an algorithm based on application of a steepest ascent heuristic to specially constructed subproblems and the classical random restart method. Computational results on three sets of standard test problems indicate that the first of these techniques outperforms the second one and is very competitive when compared to other heuristic algorithms.

1. Indroduction

Given an undirected graph $G = (V, E)$ with vertex set $V = \{1, \ldots, n\}$, edge set $E \subseteq V \times V$ and weights w_{ij} associated with the edges $(i, j) \in E$, the *MAX-CUT problem* asks for a subset of vertices V_1 such that the weight of the cut $(V_1, V_2 = V \setminus V_1)$ given by $w(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij}$ is maximized. Introducing binary variables $x_i, i \in V$, the problem can be stated as follows

$$\text{maximize } F(x) = \sum_{(i,j) \in E} w_{ij} (x_i - x_j)^2 \quad (1)$$

$$\text{subject to } x_i \in \{0,1\}, i \in V. \quad (2)$$

In (1), (2) a partition (V_1, V_2) is represented by $x = (x_1, \ldots, x_n)$ which is the incidence vector of the subset V_1, that is, $x_i = 1$ if and only if $i \in V_1$.

The MAX-CUT problem is of considerable practical significance. It has a large number of applications, the most known of which are found in design automation [1, 4, 5] and statistical physics [1, 6].

The MAX-CUT problem is NP-hard even in the case when $w_{ij} = 1$ for each edge $(i, j) \in E$. Therefore, exact algorithms require exponential time in the worst case and in practice can solve only small or at most moderately sized MAX-CUT instances. For larger graphs only heuristic techniques are applicable. Such algorithms for the MAX-CUT problem include a projected gradient algorithm [2], a rank-2 relaxation heuristic [3], a pure and hybrid GRASP [7], a pure and hybrid variable neighborhood search algorithm [7], and a combination of the rank-2 heuristic with path-relinking [8]. Goemans and Williamson [10] proposed a randomized algorithm based on solving a semidefinite programming relaxation of the MAX-CUT problem. If all edge weights w_{ij} are positive, then their algorithm produces a solution whose expected value is within a factor of 0.87856 of the optimum value. Many important research results on the MAX-CUT problem can be found in a survey [16].

The model (1), (2) can be viewed as a partial case of the unconstrained binary quadratic optimization problem:

$$\text{maximize } f(x) = \sum_{(i,j) \in E} c_{ij} x_i x_j + \sum_{i=1}^{n} c_i x_i \quad (3)$$

subject to (2); here $c_{ij} = -2w_{ij}, (i, j) \in E$, and $c_i, i \in V$, is the sum of the weights of the edges incident to i (we assume in the rest of the paper that c_{ij} and c_{ji} denote the same object – coefficient in (3) corresponding to the edge (i, j)).

In [14] several multistart tabu search strategies for (3), (2) were experimentally compared. The best performance was shown by a multistart strategy based on application of a deterministic heuristic to specially constructed subproblems (projections) of (3), (2). In this paper we adopt this algorithm for solving the MAX-CUT problem. For comparison purposes, we also investigate the classical random restart procedure with tabu search in the local improvement phase. The basic concepts of tabu search can be found, for example, in [9].

The paper is organized as follows. In Section 2, we present the algorithms for (1), (2). In Section 3, we report the results of experiments. In Section 4, we conclude with a few final remarks.

2. Algorithms

In this section we briefly describe two multistart tabu search algorithms adopted for solving the MAX-CUT problem. The algorithms deal with the transformed instances of (3), (2). The new instance is constructed by mapping the current solution $x = (x_1, \ldots, x_n)$ to (3), (2) to the zero vector. This operation amounts to replacing x_i in (3) with $1 - x_i$ for each i such that $x_i = 1$. Let c'_{ij}, c'_i stand for the coefficients of the objective function obtained after this mapping. It is easy to see that $c'_{ij} = c_{ij}\left(1 - 2(x_i - x_j)^2\right)$,

$$c'_i = (1 - 2x_i)\left(c_i + \sum_{j,(i,j) \in E, x_j = 1} c_{ij}\right).$$

The constant term of the new objective function f' is equal to $f(x)$. When dealing with the transformed instance this term always can be released.

The first algorithm generates new starting points by fixing values of some variables at 0 and then applying a steepest ascent procedure to the projection of the problem constructed by removing the fixed variables. The algorithm, named MST, can be described as follows.

MST

1. Randomly generate an 0–1 vector $x = (x_1, \ldots, x_n)$. Map x to the zero vector getting $c'_{ij}, (i,j) \in E$, and $c'_i, i \in V$. Set $x^* := x, f^* := f(x)$.
2. Apply tabu search procedure TS $\left(x, x^*, f^*, \overline{b}_1\right)$.
3. While stopping criterion is not satisfied repeat the following steps.
   3.1. Select a subset of variables $X = \left\{x_i \middle| i \in V^*\right\}$of size $n' = \lfloor \alpha n \rfloor$.
   3.2. Apply the steepest ascent procedure to the subproblem defined by X (it is obtained by fixing each $x_i \notin X$ at 0). Let x' be a solution returned by it.
   3.3. Set $x_i := 1 - x_i$ for each $i \in V^*$ such that $x'_i = 1$. Map x to the zero vector getting updated $c'_{ij}, (i,j) \in E$, and $c'_i, i \in V$.
   3.4. Apply TS $\left(x, x^*, f^*, \overline{b}_2\right)$.
4. Stop with the solution x^* of value f^*.

In Step 3.1 of this algorithm the variables x_i are included into X (and their indices into V^*) sequentially. This process is randomized by assigning to x_i the probabilities proportional to the attractiveness measure e_i calculated as follows: $e_i = 1 - d_i / d_{\min}$ if $d_i \le 0$ and $d_{\min} < 0$, $e_i = 0$ if $d_i = d_{\min} = 0$, and $e_i = 1 + \lambda d_i / d_{\max}$ if $d_i > 0$, where λ is some tuning factor, $d_{\min} = \min_{i \in V \setminus V^*} d_i$, $d_{\max} = \max_{i \in V \setminus V^*} d_i$, and d_i is the increase (or decrease if $d_i < 0$) in the value of f' that can be gained as a result of fixing x_i at 1. Initially, $d_i = c'_i, i \in V$. After a vertex $k \in V \setminus V^*$ has been moved to V^*, d_i is updated for each vertex $i \in V \setminus V^*$ adjacent to k by setting $d_i := d_i + c'_{ik}$. It is clear that the vertices with large d_i are more attractive. The experiments showed that the value of λ should be sufficiently large, too. For example, $\lambda = 500$ is apt. However, we observed that for the MAX-CUT problem slightly better results are obtained when λ is drawn randomly from some interval $[h_1, h_2]$. Such a strategy increases the level of diversification while constructing new starting points for TS. In particular, we have taken $h_1 = 5, h_2 = 5000$. Another parameter used in Step 3.1 is coefficient α controlling the size of X. In the experiments, we set α to 0.4.

In Step 3.2 of MST we apply a constructive algorithm for (3), (2) described in [13]. The idea of this algorithm is to make a steepest ascent from the center of an n' – dimensional unit cube (0.5, 0.5, …, 0.5) to some its vertex (0–1 vector) by fixing one variable at 0 or 1 at each step of this climb. The algorithm is applied to the transformed subproblem, that is, to the one of type (3), (2) with c'_{ij} and c'_i instead of c_{ij} and c_i.

The loop 3.1–3.4 is executed until a selected stopping criterion is met. In our implementation we used a stopping rule based on the CPU clock. The number of repetitions of this loop, of course, depends on the time taken by a run of tabu search procedure TS. This time interval is controlled by the last parameter submitted to TS – coefficient $\overline{b}$ used to bound the number of tabu search iterations. The overall algorithm has a warm-up phase (Step 2) in which TS is allowed to run longer $\left(\overline{b}_1 > \overline{b}_2\right)$ than in Step 3.4. The tabu search procedure for (3), (2) can be formally stated as follows.

TS $\left(x, x^*, f^*, \bar{b}\right)$

1. Set $b := 0, \bar{f} := f(x)$, tabu value $T_i := 0, i \in V$.
2. Set $L := -\infty, \gamma := 0$.
3. For $k = 1,\ldots,n$ do
   3.1. Increment b by 1.
   3.2. If $\bar{f} + c'_k > f^*$, then set $q := k, \gamma := 1$ and go to 4.
   3.3. If $T_k > 0$, then perform 3.1 for next *k*.
   3.4. If $c'_k > L$, then set $L := c'_k, q := k, a := 1$. Otherwise check whether $c'_k = L$. If so, then increment a by 1, randomly select a number $\varsigma \in [0,1]$ and, if $\varsigma \le 1/a$, set $q := k$.
4. Set $x_q := 1 - x_q, \bar{f} := \bar{f} + c'_q$. Update c'_{ij}, $(i, j) \in E$, and $c'_i, i \in V$, to keep x to be corresponding in the transformed problem instance to the zero vector. If $\gamma = 0$, then go to 6. Otherwise proceed to 5.
5. Apply a local search procedure to x. It returns possibly improved solution x and value improvement f_{local} (if $f_{\text{local}} > 0$, then c'_{ij}, c'_i are also updated). Set $\bar{f} := \bar{f} + f_{\text{local}}, x^* := x, f^* := \bar{f}$.
6. Decrement T_i by 1 for each positive $T_i, i \in V$. Set $T_q := T$, where T is the tabu tenure value. If $b < \bar{b}n$, then go to 2. Otherwise return.

The local search procedure invoked in Step 5 of the above algorithm returns a solution that is locally optimal in the neighborhood

$$N_1(x) = \left\{ (x'_1,\ldots,x'_n) \,\middle|\, \sum_{i=1}^{n} |x'_i - x_i| \le 1 \right\}.$$

This procedure like TS itself works with the transformed problem, too. So, if no variable is flipped in its value, then $f_{\text{local}} = 0$ is returned.

Besides x, x^*, f^* and $\bar{b}$ listed explicitly TS also has an additional parameter, namely, the tabu tenure value T. We run TS on the MAX-CUT problem instances with $T = 20$. The same value was used in [14] when dealing with the unconstrained binary quadratic optimization problem.

In the experiments we also tried the classical random restart algorithm formulated for the MAX-CUT problem given in the form (3), (2). This method consists of two steps executed repeatedly: generation of random starting solution and invocation of TS for this solution. In the next section we will refer to it as RRT ("Random Restart Tabu"). We believe that it is a good practice to compare any more elaborated multistart method against this traditional multistart approach.

3. Experimental results

The main purpose of experimentation was to investigate the capabilities of tabu search in solving instances of the MAX-CUT problem and to compare the obtained results with those reported in the literature.

The algorithms we have presented in the previous section were coded in the C programming language and the tests were carried out on a Pentium III 800 PC. We run MST with $\bar{b}_1 = 25000$, $\bar{b}_2 = 10000$ and RRT with $\bar{b} = 10000$.

In the first experiment, we tried MST and RRT on problem instances G1, G2, G3, G11,…, G16, G22, G23, G24, G32,…, G37, G43, G44, and G45 created by Helmberg and Rendl [12] and used by several authors including [3, 7, 8] for testing their algorithms. The solution values and average computation times for MST and RRT on these instances are listed in Tables 1 and 2. For comparison purposes, Table 1 also includes the results obtained with most successful algorithms described in the literature: variable neighborhood search with forward path-relinking *vnspr* presented by Festa, Pardalos, Resende and Ribeiro [7], rank-2 relaxation heuristic *circut* developed by Burer, Monteiro and Zhang [3], and a hybrid of *circut* and path-relinking *circut+pr* proposed by Festa and Resende [8]. The data (cut value in one run) for *vnspr* (third column) are taken from [7] and the data (best cut value in 10 runs) for *circut* and *circut+pr* (fourth and fifth columns) from [8]. The first two columns of each of Tables 1 and 2 give the problem (graph) identifier and the number of the vertices of the graph. The last two columns of Table 1 display the value of the best solution obtained from 10 runs of RRT and MST, respectively. The columns under heading "RRT" and "MST" in Table 2 list for each graph the average cut value and the average time taken to first find a solution that is best in the run. Each run was limited to 1800 seconds for a graph of order 800 and to 3600 seconds for a graph of order 1000 or 2000.

By analyzing the results in Tables 1 and 2, we find that MST for this class of instances, in general, performs better than RRT, especially when comparison is based on the best solutions produced by these techniques. Each of them was able to find for some graphs a cut of weight larger than that known in the literature. Specifically, MST has improved the best known values for G14, G15, G16, G22 and G44 and RRT for G44 and G45 (all these values in Table 1 are indicated in bold face). We can also see from Table 1 that the best results are obtained by *circut+pr*.

Table 1. Best solutions found by different techniques for Helmberg and Rendl instances

Problem	*n*	*vnspr*	*circut*	*circut+pr*	RRT	MST
G1	800	11621	11624	11624	11624	11624
G2	800	11615	11620	11620	11620	11620
G3	800	11622	11622	11622	11622	11622
G11	800	564	558	564	564	562
G12	800	556	554	556	556	552
G13	800	580	582	582	580	576
G14	800	3055	3061	3061	3042	**3063**
G15	800	3043	3049	3049	3024	**3050**
G16	800	3043	–	–	3026	**3052**
G22	2000	13295	13354	13355	13235	**13358**
G23	2000	13290	13354	13338	13246	13329
G24	2000	13276	13329	13331	13241	13327
G32	2000	1396	1396	1402	1384	1392
G33	2000	1376	1368	1372	1358	1368
G34	2000	1372	1372	1376	1362	1368
G35	2000	7635	7672	7672	7590	7672
G36	2000	7632	7669	7670	7577	7669
G37	2000	7643	7680	7681	7589	7675
G43	1000	6659	6660	6660	6660	6660
G44	1000	6642	6649	6649	**6650**	**6650**
G45	1000	6646	6653	6653	**6654**	6650

Table 2. Average solutions found by MST and RRT and average time (in seconds) to the best solution in the run for Helmberg and Rendl instances

Problem	*n*	RRT		MST	
		value	time	value	time
G1	800	11624	15	11610	147
G2	800	11620	180	11607	195
G3	800	11622	54	11611	278
G11	800	564	819	558	213
G12	800	554	736	547	181
G13	800	579	640	571	414
G14	800	3037	785	3059	787
G15	800	3018	581	3047	1109
G16	800	3022	683	3048	916
G22	2000	13221	2039	13306	1519
G23	2000	13224	2162	13302	1634
G24	2000	13223	1381	13308	1824
G32	2000	1380	1498	1385	1290
G33	2000	1355	1054	1357	812
G34	2000	1359	1803	1359	1324
G35	2000	7582	1200	7668	2863
G36	2000	7571	2557	7661	2578
G37	2000	7582	1974	7669	2384
G43	1000	6660	845	6648	733
G44	1000	6650	1484	6639	478
G45	1000	6653	800	6640	596

However, computation times for *circut+pr* (as reported in [8]) were very large: for G1 – G3, for example, about 36000 seconds on an SGI Challenge with a 196 MHz R10000 processor. Slightly inferior solutions are produced by *circut* which, on the other hand, is incomparably faster than *circut+pr*. Comparing MST and *vnspr*, we can see that our algorithm in most cases found cuts of larger weight than *vnspr*. In general, we can conclude that there is no clear winner among the compared algorithms. Even the random restart method RRT sometimes performs superbly. It simply beats other competitors on the graph clusters G1–G3 and G43–G44 by finding a solution of the best known value in almost each run (10 times for G1, G2 and G3, 9 times for G43 and G44, and 7 times for G45; average values are given in the third column of Table 2).

In the second experiment, we considered ten MAX-CUT problem instances sg3dl101000,…, sg3dl1010000 of size 1000 and ten instances sg3dl141000,…, sg3dl1410000 of size 2744 used by Burer, Monteiro and Zhang [3]. These instances (graphs) are constructed from cubic lattices modeling Ising spin glasses (see [3] for details). In Table 3 (the last two columns) we give for each graph the value of the best solution found by each of the algorithms RRT and MST in 5 runs. Each run was limited to 3600 seconds for the first ten (smaller) graphs and to 7200 seconds for the ten larger graphs. We also include in this table the results from the literature: from [7] for *circut* and *vnspr* and from [3] for the algorithm proposed by Hartmann [11] (the column under heading "H2"). The latter algorithm focuses on finding the groundstates of Ising spin glasses that can be embedded as square or cubic lattices in two or three dimensions, respectively. Since the used instances are of such type it is not a surprise that the approach of Hartmann produces significantly better solutions than any of the other competitors. However, such a good performance is achieved at the expense of very large computation times: for sg3dl14 series about 33000 seconds per instance on SGI Origin 2000 machine (see [3] for the exact timing of H2 and for a more detailed characterization of the computer used).

As it can be seen from Table 3 MST again produced better cuts than RRT. The difference between cut values especially large is for sg3dl14 series of graphs. Our algorithm MST compares favourably also with *circut* and *vnspr*. Compared with *circut* for sg3dl14 series, for example, MST found better cuts in 48 runs (out of 50) and tied in 2 runs. In comparison with *vnspr*, MST improved in 47 runs, tied in 1 run, and produced inferior solutions in only two cases.

The structure of Table 4 is very similar to that of Table 2. We can see from it that the average time taken to first find a solution that is best in the run for RRT is noticeably smaller than for MST. This was not a case for Helmberg and Rendl graphs.

Table 3. Solutions found by different techniques for Burer, Monteiro and Zhang instances

Problem	*circut*	*vnspr*	H2	RRT	MST
sg3dl101000	880	892	896	892	896
sg3dl102000	892	900	900	898	900
sg3dl103000	882	884	892	886	888
sg3dl104000	894	896	898	896	896
sg3dl105000	882	882	886	884	884
sg3dl106000	886	880	888	884	888
sg3dl107000	894	896	900	898	898
sg3dl108000	874	880	882	880	880
sg3dl109000	890	898	902	900	902
sg3dl1010000	886	890	894	890	892
sg3dl141000	2410	2416	2446	2378	2438
sg3dl142000	2416	2416	2458	2394	2448
sg3dl143000	2408	2406	2442	2394	2434
sg3dl144000	2414	2418	2450	2390	2436
sg3dl145000	2406	2416	2446	2380	2432
sg3dl146000	2412	2420	2450	2394	2440
sg3dl147000	2410	2404	2444	2384	2434
sg3dl148000	2418	2418	2446	2386	2434
sg3dl149000	2388	2384	2424	2362	2416
sg3dl1410000	2420	2422	2458	2402	2450

Table 4. Average solutions found by MST and RRT and average time (in seconds) to the best solution in the run for Burer, Monteiro and Zhang instances

Problem	RRT		MST	
	value	time	value	time
sg3dl101000	888.4	961	889.6	1755
sg3dl102000	896.8	1438	896.8	2208
sg3dl103000	884.4	578	883.2	1616
sg3dl104000	894.4	1677	892.4	913
sg3dl105000	881.6	2163	881.2	1722
sg3dl106000	882.4	1735	883.6	1808
sg3dl107000	896.0	1619	894.8	2203
sg3dl108000	877.2	511	878.4	712
sg3dl109000	896.4	1472	890.8	1391
sg3dl1010000	888.4	1311	888.0	297
sg3dl141000	2377.6	4833	2425.2	5265
sg3dl142000	2392.0	3035	2436.4	4645
sg3dl143000	2382.0	4286	2422.4	3878
sg3dl144000	2384.8	3279	2430.4	3665
sg3dl145000	2376.4	3701	2424.4	5871
sg3dl146000	2388.4	3614	2429.6	2760
sg3dl147000	2377.2	2146	2420.4	5405
sg3dl148000	2382.0	2864	2424.8	5726
sg3dl149000	2360.8	4258	2406.4	4784
sg3dl1410000	2392.8	3869	2433.2	5099

Table 5. Solutions found by different techniques for the torus problems

Problem	*n*	*circut*	SA	RRT	MST
pm3-8-50	512	454	458	458	458
pm3-15-50	3375	2964	3016	2930	3000
g3-8	512	41684814	39111654	40043061	41684814
g3-15	3375	281029888	260202525	251918092	283206561

Table 6. Average solutions found by MST and RRT and average time (in seconds) to the best solution in the run for the torus problems

Problem	RRT		MST	
	value	time	value	time
pm3-8-50	458.0	745	456.0	785
pm3-15-50	2924.4	3611	2992.4	4172
g3-8	39914366.0	639	41647774.4	841
g3-15	250725220.6	4458	282541878.0	2764

The last experiment was conducted on a set of test problems taken from the DIMACS library of semidefinite-quadratic-linear programs [15]. This set contains four instances of MAX-CUT, called the torus problems, which originated from the Ising model of spin glasses in physics (see [15] for details). We run MST and RRT on each instance 5 times for 1800 seconds in the case of pm3-8-50 and g3-8 and for 7200 seconds in the case of pm3-15-50 and g3-15. The results are reported in Tables 5 and 6. Table 5 also includes the results from the literature: from [3] for *circut* and from [15] for an implementation of the simulated annealing algorithm SA. Tables 5 and 6 clearly show that MST is quite effective in obtaining high-quality solutions for the torus set. In particular, for g3-8 MST was able two times (out of 5) to find a cut of value 41684814 that is known to be optimal [3]. For pm3-8-50 the best performance is demonstrated by RRT. This algorithm can find cuts of weight 458 (which is the best known value), perhaps, constantly within the allotted half hour. For pm3-15-50 the best cut is produced by SA. For g3-15 a rather good solution is found by MST. We believe that significantly better solutions for this instance can be obtained in longer runs of MST.

4. Conclusions

In this paper we presented two multistart tabu search implementations for the MAX-CUT problem. The results of experiments show that the algorithm based on construction of starting points using a one-pass heuristic for (3), (2), in general, outperforms the random restart method. The algorithm can quickly find solutions that are competitive with those found by most successful algorithms described in the literature. For 6 benchmark graphs the solutions of weight larger than the best known value were produced.

References

[1] **F. Barahona, M. Grötschel, M. Jünger, G. Reinelt.** An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 1988, 36, 493–513.

[2] **S. Burer, R.D.C. Monteiro.** A projected gradient algorithm for solving the maxcut SDP relaxation. *Optimization Methods and Software*, 2001, 15, 175–200.

[3] **S. Burer, R.D.C. Monteiro, Y. Zhang.** Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. *SIAM J. on Optimization*, 2002, 12, 503–521.

[4] **K.C. Chang, D.H.-C. Du.** Efficient algorithms for layer assignment problem. *IEEE Trans. on Computer – Aided Design of Integrated Circuits and Systems*, 1987, 6, 67–78.

[5] **R. Chen, Y. Kajitani, S. Chan.** A graph-theoretic via minimization algorithm for two-layer printed circuit boards. *IEEE Trans. on Circuits and Systems*, 1983, 30, 284–299.

[6] **C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, G. Rinaldi**. Exact ground states of Ising spin glasses: new experimental results with a branch and cut algorithm. *Journal of Statistical Physics*, 1995, 80, 487–496.

[7] **P. Festa, P.M. Pardalos, M.G.C. Resende, C.C. Ribeiro.** Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 2002, 17, 1033–1058.

[8] **P. Festa, M.G.C. Resende.** CIRCUT+PR: a rank-2 heuristic with path-relinking for MAX-CUT. *Extended abstracts of the Fifth Metaheuristics International Conference MIC2003*, *Kyoto, Japan*, 2003.

[9] **F. Glover, M. Laguna.** Tabu search. *Kluwer Academic Publishers, Hingham*, *MA, USA*, 1997.

[10] **M.X. Goemans, D.P. Williamson.** Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM,* 1995, 42, 1115–1145.

[11] **A.K. Hartmann.** Cluster-exact approximation of spin glass groundstates. *Working paper, Institut für Theoretische Physik, Universität Heidelberg*, *Heidelberg*, July 1998.

[12] **C. Helmberg, F. Rendl.** A spectral bundle method for semidefinite programming. *SIAM J. on Optimization*, 2000, 10, 673–696.

[13] **G. Palubeckis.** Heuristics with a worst-case bound for unconstrained quadratic 0–1 programming. *Informatica*, 1992, 3, 225–240.

[14] **G. Palubeckis.** Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, in press.

[15] **G. Pataki, S. Schmieta.** The DIMACS library of semidefinite-quadratic-linear programs. *Technical report, Computational Optimization Research Center*, *Columbia University*, 2002.

[16] **S. Poljak, Z. Tuza.** Maximum cuts and large bipartite subgraphs. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1995, 20, 181–244.